

## DATA TRANSFER CONTROL METHOD FOR PARALLEL COMPUTER

**Publication number: JP9146903**

**Publication date:** 1997-06-06

**Inventor:** SAGAWA NOBUTOSHI; SUKEGAWA NAONOBU

**Applicant:** HITACHI LTD

**Classification:**

**- international:** **G06F13/00; G06F15/16; G06F13/00; G06F15/16;**  
(IPC1-7): G06F15/16; G06F13/00

**- european:**

**Application number: JP19950307111 19951127**

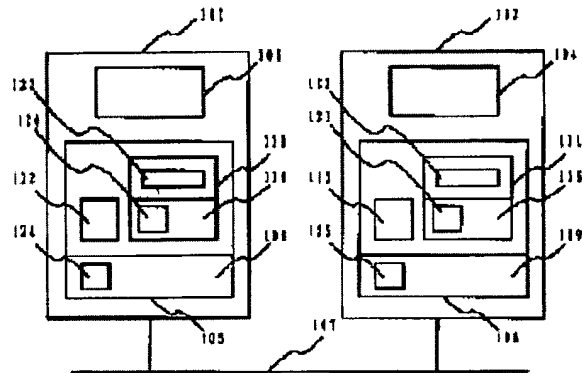
**Priority number(s):** JP19950307111 19951127

**Report a data error here**

## Abstract of JP9146903

**PROBLEM TO BE SOLVED:** To transmit and receive messages safely on a parallel computer by using remote memory transfer.

**SOLUTION:** When messages are transmitted to and received from the parallel computer by remote memory transfer, the respective element computers are provided with an arbitrary number of memory areas 122 and 123 that can be accessed remotely and their management areas 124 and 125, and flags for remote access inhibition are provided in the management areas for the respective memory areas, thereby optionally allowing the areas to be accessed remotely or inhibiting them from being accessed remotely. Consequently, data are prevented from being overwritten to the same memory area owing to multiple writing and illegal data are prevented from being read out owing to the repetition of writing to and reading from the same memory area.



Data supplied from the **esp@cenet** database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-146903

(43) 公開日 平成9年(1997)6月6日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/16	4 7 0		G 0 6 F 15/16	4 7 0 D
13/00	3 5 5		13/00	3 5 5

審査請求 未請求 請求項の数 3 O L (全 11 頁)

(21) 出願番号 特願平7-307111

(22) 出願日 平成7年(1995)11月27日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 佐川 暢俊

東京都国分寺市東壱ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(72) 発明者 助川 直伸

東京都国分寺市東壱ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(74) 代理人 弁理士 小川 勝男

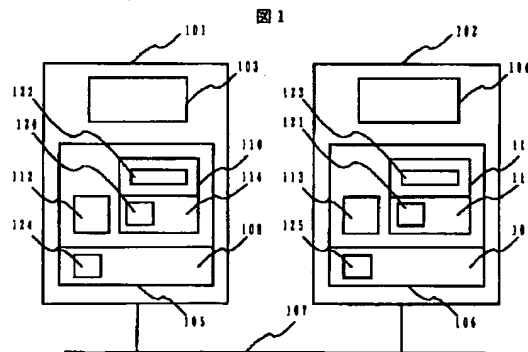
(54) 【発明の名称】 並列計算機におけるデータ転送制御方法

(57) 【要約】

【課題】 並列計算機上でリモートメモリ転送を用いたメッセージの送受信を安全に行なう。

【解決手段】 並列計算機上でリモートメモリ転送によりメッセージを送受信する際に、各要素計算機上にリモートアクセス可能な任意個数のメモリ領域(122, 123)とその管理領域(124, 125)を設け、各々のメモリ領域について管理領域上にリモートアクセス禁止のフラグを設けることにより、その領域に対してリモートアクセスを任意に許可あるいは禁止できるようにする。

【効果】 同一メモリ領域への複数の書き込みによりデータがオーバーライトされること、および同一メモリ領域への書き込みと読み出しの重複により不正なデータが読み出されることを防止できる。



## 【特許請求の範囲】

【請求項1】複数台の計算機を通信路によって結合してなり、該計算機がデータ転送先の計算機のメモリ領域を指定してデータ転送を行う並列計算機におけるデータ転送制御方法であって、

各々の計算上で、他の計算機上で実行されているプログラムからデータの読み出し、書き込みを行なうことのできるメモリ領域を、自計算機上で実行されているプログラムから任意個数指定し、

該指定された複数のメモリ領域の各々について、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みが禁止されていることを示す情報を保持し、自計算機上で実行されているプログラムから該情報を制御することにより、該メモリ領域へのデータの読みだしおよび書き込みを禁止することを特徴とする並列計算機におけるデータ転送制御方法。

【請求項2】該指定された複数のメモリ領域のうち、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みを禁止された該メモリ領域に対し、他の計算機で実行されているプログラムからデータの読み出しが要求された場合、該他の計算機上のプログラムに対しエラーを通知することを特徴とする請求項1記載の並列計算機におけるデータ転送制御方法。

【請求項3】該指定された複数のメモリ領域のうち、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みを禁止された該メモリ領域に対し、他の計算機上で実行されているプログラムからデータの書き込みが要求された場合、該データを破棄し、該データを破棄した計算機において、該データの破棄を、該計算機上で実行されているプログラムに対して通知することを特徴とする請求項1記載の並列計算機におけるデータ転送制御方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する利用分野】本発明は複数の要素計算機（プロセッシングユニット、以下PU）を通信網によって結合した並列計算機におけるPU間のデータ通信に係わり、特にデータ通信が相手PUの介入なしに行なわれるリモートメモリへのデータ転送時のデータの安全性の確保に関する。

## 【0002】

【従来の技術】並列計算機は、複数のPUを通信網によって結合し、それらを同時に稼働させることによって処理速度を向上させる。本発明では特に、各PUがそれに付随するメモリ空間のみをアクセスすることができる分散メモリ型の並列計算機を対象とする。分散メモリ型並列計算機では、他のPUのメモリ上にあるデータを直接アクセスすることはできない。データが必要となる度に送受信を行なってそのデータを自PUに移動する必要がある。

【0003】分散メモリ型並列計算機では、PU間のデータのやりとりをすべてプログラム中に記述する必要がある。ここで、PU間で受け渡されるデータをメッセージと呼ぶ。並列計算機用プログラムでは、他のPUで必要となるデータが自PUのメモリ上にある場合にはそれをあらかじめ送信し、他のPUのメモリ上にあるデータを自PUが必要とする場合にはあらかじめ受信しておくような指示を各PUのプログラム中に明示的に記述する必要がある。多くの並列計算機システムでは、このようなPU間のメッセージの送受信をサポートする目的で、メッセージパッシングライブラリと呼ばれる関数（あるいはサブルーチン）群があらかじめ用意されており、通信は「C」や「FORTRAN」などのプログラムからの関数コールとして記述できるようになっている。メッセージパッシングライブラリの中には、異なる並列計算機ハードウェア上にインプリメントされ、事実上の標準としての通信環境を提供するものも現われている。米国Oak Ridge National Laboratoryで開発されたPVMや、近年標準化が進められているMPI (Message Passing Interface) はその例である。これらの通信ライブラリをコールすることにより書かれた並列プログラムは、異なる並列計算機上でも再コンパイルのみで動作させる可能性（可搬性）が高い。

【0004】通信ライブラリでPU間のメッセージの受け渡しを行なうには、2種類の方式が知られている。第1は、送信側PUでメッセージの送信関数をコールし、受信側PUでそれに対応するメッセージの受信関数をコールし、これらの中でメッセージを送受信する方式である。送信関数より受信関数が先にコールされた場合には受信関数はデータの到着までブロック（停止）し、送信関数が先にコールされた場合には受信関数の発行までブロックするか、メッセージがシステム内にバッファリングされるのが一般的である。これはsend/receive方式と呼ばれる。第2は、送信側PUでメッセージの送信関数をコールしただけで（対応する受信関数の発行なしに）メッセージを相手PUに書き込み、あるいは受信関数のコールにより（対応する送信関数の発行なしに）データを相手PUから読み込む方式である。これはput/get方式、あるいはリモートメモリコピー方式と呼ばれ、送信関数はput関数、受信関数はget関数と呼ばれる。

【0005】上に挙げた標準的なメッセージパッシングライブラリのうちPVMではsend/receive方式による通信のみをサポートしている。MPIもsend/receive方式のみをサポートしているが、時期バージョンとして現在検討が進められているMPI-2ではこれに加えてput/get方式による通信も取り入れる計画がある。

【0006】send/receive方式は、送信

側、受信側のPUがそれぞれに送信、受信関数を発行して明示的にメッセージを受け渡すため、利用者はどのタイミングでメッセージが自PUのどのメモリ領域に送受信されるかを制御することができる。しかし、利用者はプログラム中で送信と受信の対応付けを常に行なわなくてはならないので、プログラムの手間は大きい。これに対し、put/get方式は送信側、受信側がそれぞれ一方的に相手PU上にあるデータを読み書きする。データが必要になった時点でput、あるいはgetの一方を発行すればよいので、プログラムの手間は少なくなる。しかし、メッセージの送受信先メモリ領域を誤った場合、あるいは2つ以上のPUがほぼ同時に同一のメモリ領域に読み書きを行なった場合には、データを破壊したり、データを上書きしたり、誤ったデータを読み出す恐れがある。

【0007】MPI-2では、送受信先メモリ領域の誤りにより利用者プログラムが破壊されることを防ぐ目的で、各PUにput/get専用のメモリ領域を確保する関数を提供している。put/get操作は、各PUであらかじめ確保された領域に対してしかメッセージの送受信を行なえないので、それ以外のメモリ領域に誤って書き出し、読み込みを行なう可能性がなくなる。

【0008】以下、メッセージバッシングライブラリMPI-2で提案されているput/get関数を用いたメッセージ通信方法の概略を説明する。ただし、説明を簡単にするために、関数仕様は一部簡略化して記述する。

【0009】MPI利用時には、まず通信に参画する全プログラム上で次のようなMPI初期化関数をコールする。通常この処理は並列プログラムの先頭で行なう。

【0010】Init()

本関数中で、メッセージバッシングライブラリは必要な初期化操作を行なう。以下に挙げる関数は、初期化関数をコールした後でのみ使用することができる。

【0011】初期化終了後put/getを用いる際には、全PU上でput/get用に専用のメモリ領域を確保するための次の関数をコールする。

【0012】Rmc\_malloc(base, size, id)

ここでbaseは確保するメモリ領域(ウィンドウ)の先頭アドレス、sizeはメモリ領域の長さである。idはウィンドウの識別子であり本関数の出力引き数である。本関数を複数回コールすることにより、ウィンドウを一つのPU上に任意個数設けることができる。本関数で生成したウィンドウには、get/put関数からセットできるカウンタが自動的に付属する。このカウンタは直接利用者がアクセスすることはできないが、後述のハンドラを用いてその値を監視することで、ウィンドウに対するput/get操作の完了を検知することができる。

【0013】他のPUのウィンドウに対するput操作

は次の関数コールにより実行する。

Put(origin\_addr, origin\_size, target\_rank, id, incr)

ここでorigin\_addr, origin\_sizeは送るべきメッセージが格納された自PUのメモリの先頭アドレスと長さである。target\_rankは送り先のPU番号である。idはputの対象となる相手PU上のウィンドウを示す。incrは、当該ウィンドウに付属するカウンタのインクリメント値を表わし、メッセージの書き込みと同時にput先ウィンドウのカウンタがここに指定した値だけインクリメントされる。

【0014】一方、get操作は次の関数コールによる。

【0015】Get(origin\_addr, origin\_size, target\_rank, id, incr)

各引数の意味はgetの場合と同様である。ただし、本関数ではメッセージが相手PUのウィンドウから読み出され、自PUのorigin\_addr以降のメモリに受信される。

【0016】put/get操作の完了は、カウンタが利用者があらかじめ指定した上限値を越えたかどうかで判定する。ここで、カウンタの上限値は利用者がウィンドウごとに次の関数をコールすることにより設定することができる。

【0017】Set\_counter\_threshold(id, count)

idはウィンドウの識別子、countは設定すべき上限値である。カウンタが上限値を超えた場合には、利用者があらかじめ登録したハンドリング関数がMPIによりコールバックされる。ハンドリング関数の設定は、次の関数コールにより行なう。

【0018】

Post\_handler(id, handler)  
idはハンドリング関数の呼び出し対象となるウィンドウの識別子、handlerはハンドリング関数へのポインタである。ハンドリング関数自体は利用者が自由に記述することができる。カウンタの上限値に1を指定し、put/getの引き数incrにも1を指定すれば、個々のput/getの終了時にハンドリング関数を呼び出すことができる。また、カウンタの上限値とincrで指定する値を適宜変更することで、複数回のput/getが終了した時点でハンドリング関数をコールすることも可能である。

【0019】以上が、MPIを用いた場合の利用者プログラムからのput/get操作方法の説明である。

【0020】

【発明が解決しようとする課題】上述のようにput/

get方式は潜在的にデータを破壊する可能性のある危険な通信方式であるにもかかわらず、MPIを含む従来のメッセージバッシングライブラリでは2つ以上のPUが同一のメモリ領域に読み書きを行なった場合に対する考慮がなされていなかった。また、put/get操作によるデータの予期しない破壊が起こった場合に、利用者からこれを検出する手段が設けられていなかった。

【0021】本発明の目的は、分散メモリ型並列計算機のメッセージバッシングライブラリにおいて、利用者がput/get操作による同一メモリ領域へのメッセージの重複した書き込みを防止することと、同一メモリ領域への重複した書き込みがあった場合にこれを検出することを可能とし、put/get方式による通信の安全性を高めることにある。

【0022】

【課題を解決するための手段】本発明は、put/get方式による送受信において、複数のPUからの同一メモリ領域へのput/getの発行によるデータの上書きや誤ったデータの読み出しを防ぐために、各PU上でput/get用に確保したメモリ領域に対して書き込み、読み出し禁止フラグを設け、そのフラグを利用者プログラムから制御可能とすることにより、利用者が当該メモリ領域への他PUからの書き込み、読み出しを禁止するようにすることによって達成される。

【0023】また、同一メモリ領域への重複した書き込み、読み出しを利用者が検出できるようにするために、書き込み、読み出し禁止フラグの立ったメモリ領域へのput/getの発行を利用者に通知するようにすることによって達成される。

【0024】

【発明の実施の形態】以下、図を参照して本発明の詳細を説明する。

【0025】まず、本発明の実装方法の具体例を図を参照して説明する。図1に本発明の全体構成図を示す。101、102はPUを示し、103、104はそれらのCPU、105、106はメモリである。107はそれらPUを結ぶ通信路である。PUの数は実際には任意であるが、ここでは説明のために2つのPUからなる並列機を示している。108、109は各PUのOS（オペレーティングシステム）である。利用者プログラムを実行する際には、各PUのメモリ上にプログラムがローディングされる（110、111）。利用者プログラムは、あらかじめ本発明のメッセージバッシングライブラリ（114、115）とリンクされている。利用者プログラム中には、他のPUからリモートメモリ転送により読み書きが可能なメモリ領域（122、123）を設けることができる。さらに、並列プログラムの実行開始と同時に、put/getデーモンプロセス（112、113）が各PUごとに起動される。メッセージバッシングライブラリ中には、put/getデーモンからの割り込

み要求によって起動される特別なルーチン（割り込みハンドラ）120、121およびput/get用のメモリ領域の管理情報を格納するウィンドウ情報テーブル（124、125）が設けられる。割り込みハンドラ中には、put要求にたいする割り込みを処理するputハンドラと、get要求に対する割り込みを処理するgetハンドラが存在する。

【0026】以上の構成要素のうち、メッセージバッシングライブラリとput/getデーモンとが本発明の特徴をなす構成要素である。以下、これら2つの構成要素について詳細に説明する。

【0027】（メッセージバッシングライブラリのインターフェイス）本発明におけるメッセージバッシングライブラリ（114、115）について説明する。メッセージバッシングライブラリは利用者の並列プログラムからコールすることにより、put/getデーモンプロセスを介してメッセージバッシングを実行するための関数群である。

【0028】以下に、メッセージバッシングライブラリの関数インターフェイスを説明する。本実施例では上述のMPI-2の仕様を基本とし、それに必要な変更を加えることによって関数インターフェイスを構築する。実際には関数名、引き数名称などは任意であり、必ずしもここで説明する仕様と同じである必要はない。

【0029】（1）Init（）

本関数のインターフェイスは上述のMPIと同様である。

【0030】（2）RMC\_Malloc（base, size, counter, counter\_success, id）

自PU上にput/getウィンドウを確保する関数である。引き数base, size, counter, idの意味は、従来の技術の項で述べたMPIの場合と同様である。counter\_successは本発明にて新たに設けられたカウンタであり、利用法は後述する。

【0031】（3）Get（origin\_addr, origin\_size, target\_rank, id, incr）

本関数インターフェイスと引き数の意味は上述のMPIの場合と同様である。

【0032】（4）Put（origin\_addr, origin\_size, target\_rank, id, incr）

本関数インターフェイスと引き数の意味は上述のMPIの場合と同様である。

【0033】（5）Set\_counter\_threshold（id, count）

本関数インターフェイスと引き数の意味は上述のMPIの場合と同様である。

【0034】(6) `Post_handler(id, count)`

本関数インターフェイスと引き数の意味は上述のMP1の場合と同様である。

【0035】(7) `Window_lock(id)`

本関数と次の`Window_unlock`関数は本発明に特徴的なものである。本関数が利用者プログラムから呼び出されると、引き数`id`で指定されたウィンドウ識別子に対応するウィンドウを`put/get`禁止とする。本操作をウィンドウのロックと呼ぶ。このウィンドウを対象とする`put/get`操作は、同じウィンドウに対して次の`Window_unlock`関数がコールされるまで行なうことができない。ロックされたウィンドウに対して`put`操作を行なった場合には、そのメッセージは破棄される。ロックされたウィンドウにたいして`get`操作を行なった場合には、メッセージは読みだされず、`get`関数の戻り値としてエラーコードが返される。

【0036】

(8) `Window_unlock(id)`

本関数は、`id`で指定されたウィンドウ識別子に対するウィンドウの`put/get`禁止を解除する。本操作をウィンドウのアンロックと呼ぶ。

【0037】(メッセージバッシングライブラリの動作)

以下、本発明のリモートメモリへのデータ転送制御方式における上記各関数の実現方法を示す。

【0038】(1) `Init`の動作

メッセージバッシングライブラリの初期化関数は、利用者プログラムからコールされると、図2に示すウィンドウ情報テーブルをライブラリのメモリ領域(図1の105, 106)上に作成する。ウィンドウ情報テーブルは、ウィンドウ識別子のカラム、ウィンドウ開始アドレスのカラム、ウィンドウ末尾アドレスのカラム、カウンタ値のカラム、成功カウンタ値のカラム、カウンタの上限値のカラムおよびウィンドウのロック状態を示すフラグ(一般的にこの情報を保持できるものであればよい)のフィールドを有する。ついで、生成されたテーブルの各カラムを初期化する。初期値には、カウンタ上限値のカラムのみ1、他のすべてのカラムは0を用いる。上述のように利用者は初期化関数を全PUから呼び出すので、ウィンドウ情報テーブルも全PU上に作成される。

【0039】(2) `Rmc_malloc`関数の動作

ウィンドウ生成関数`Rmc_malloc`がコールされた場合の動作を図3に示す。まず、関数の引き数として与えられたウィンドウの先頭アドレス`buff`と長さ`size`を用いてウィンドウ用メモリを確保する(301)。ウィンドウ情報テーブルの`id`のカラムをスキャンし、`id`が0(初期値)であるカラムを見つける(302)。そのカラムの行番号をウィンドウの識別子として採用し、対応するカラムに格納する(303)。次い

で、ウィンドウの先頭アドレスのカラムに`buff`で与えられたアドレスを格納し、ウィンドウ末尾アドレスに`buff+size`の値を格納する(304, 305)。

【0040】カウンタ値、カウンタ最大値、ロック状態フラグは初期値のまま変化させない。

【0041】(3) `Put`関数の動作

次に、`Put`関数がコールされた場合の動作を説明する。図4に処理の流れを示す。まず、相手PU上の`put/get`デーモンに対し、`put`要求が出されたことを通知する短いメッセージ(`put`要求)を送り、次いで関数の引き数として与えられた送信先ウィンドウの識別子とインクリメント値を、同じく引き数`target_rank`中で指定された相手PU上の`put/get`デーモンに送信する(401)。( `put/get`デーモンの動作は後述する。) この送信は、通常の`send`関数を用いて行なう。送信後、同デーモンからの認識信号(`ack`)を待ち、ブロックする(402)。 `ack`が到着したならば、同デーモンに対して、引き数の`origin_addr`および`origin_size`で示される送信バッファの内容を送信する(403)。この送信も`send`関数によって行なう。送信バッファの内容がすべて送出されたならば、利用者プログラムへリターンする。

【0042】(4) `Get`関数の動作

同様に、`Get`関数がコールされた場合の動作を図5を用いて説明する。まず、相手PU上の`put/get`デーモンに対し、`put`要求が出されたことを通知する短いメッセージ(`get`要求)を送り、次いで関数の引き数として与えられた送信先ウィンドウの識別子とインクリメント値を、同じく引き数`target_rank`中で指定された相手PU上の`put/get`デーモンに送信する(501)。この送信は、通常の`send`関数を用いて行なう。送信後、同デーモンを対象として`receive`関数を発行しデータの到着を待つ。受け取ったメッセージは、引き数の`origin_addr`および`origin_size`で指定された受信バッファに格納する(502)。受信バッファへ内容がすべて格納されたならば、利用者プログラムへリターンする。

【0043】(5) `Set_counter_threshold`関数の動作

本関数がコールされると、自PUのウィンドウ情報テーブル(図2)から引き数で指定された識別子を持つ行をサーチし、その行のカウンタ上限値のカラムに引き数で指定されたカウンタ上限値を格納する。

【0044】(6) `Post_handler`関数の動作  
本関数がコールされると、自PUのウィンドウ情報テーブル(図2)から引き数で指定された識別子を持つ行をサーチし、その行のハンドラのカラムに引き数で指定されたハンドリング関数のポインタを格納する。

【0045】(7) Window\_lock関数の動作  
本関数がコールされると、自PUのウィンドウ情報テーブル(図2)から引き数で指定された識別子を持つ行をサーチし、その行のロック状態フラグ(書き込み、読み出し禁止フラグ)をonにする。

【0046】(8) Window\_unlock関数の動作

本関数がコールされると、自PUのウィンドウ情報テーブル(図2)から引き数で指定された識別子を持つ行をサーチし、その行のロック状態フラグ(書き込み、読み出し禁止フラグ)をoffにする。

【0047】以上が本発明のリモートメモリ転送制御方式における各関数の実現方法の例である。

【0048】(9) putハンドラの動作

メッセージバッシングライブラリ中には、利用者からコールされる上記の関数群以外に、put/getデーモンプロセスからの割り込みによって起動されるput/getハンドラが存在する。このうちputハンドラの動作を図7を参照して説明する。割り込みを受け付けると、put/getデーモンプロセスからの受信待ち状態に移行し、ウィンドウ識別子とカウンタのインクリメント値を受け取る(701)。識別子を受け取ると、それを用いて自PU上のウィンドウ情報テーブル(図2)をサーチし、対応するカウンタを受け取ったインクリメント値にしたがってインクリメントする。また、ロックフラグのカラムを調べて当該ウィンドウがロック状態にあるか否かを調べる(704)。ロックされていない場合には、put/getデーモンプロセスにackを返し(702)、続いてput/getデーモンプロセスから送信されてくるメッセージ本体を受け取り、ウィンドウに格納する(703)。また、対応する成功カウンタの値をインクリメント値にしたがってインクリメントする(706)。ロックされていた場合にはput/getデーモンプロセスにnackを返す(705)。最後にカウンタ値とカウンタ上限値とを比較し、カウンタ値が等しいか大きくなっていればカウンタをクリアして対応する利用者定義のハンドラに制御を移す(707)。

(10) getハンドラの動作

getハンドラの動作を図8を参照して説明する。割り込みを受け付けると、put/getデーモンプロセスからの受信待ち状態に移行し、ウィンドウ識別子を受け取る(801)。識別子を受け取ると、それを用いて自PU上のウィンドウ情報テーブル(図2)をサーチし、ロックフラグのカラムを調べて当該ウィンドウがロック状態にあるか否かを調べる(804)。ロックされていない場合には、put/getデーモンプロセスにackを返し(802)、続いてウィンドウに格納されているメッセージ本体をput/getデーモンプロセスに送信する(803)。ロックされていた場合にはput

/getデーモンプロセスにnackを返す(805)。最後にカウンタ値とカウンタ上限値とを比較し、カウンタ値が等しいか大きくなっていればカウンタをクリアして対応する利用者定義のハンドラに制御を移す(807)。

【0049】(11) ロックされているウィンドウに対するput/getの検出

本発明では、何等かの原因でロックされているウィンドウにメッセージがput/getされた場合、これを行うようにして検出することができる。

【0050】get関数に対しては、get先のgetハンドラからnackが返された場合、put/getデーモンプロセスはget関数発行元にたいしてエラーコードを返す(619)。したがって、利用者はget関数のリターン値がエラーコードであった場合には、ロックされたウィンドウに対してgetが発行されたことを検知できる。

【0051】put関数は、相手PUに対してメッセージを送信した後直ちにリターンするので、リターンコードによってロックされたウィンドウに書き込みを行なおうとしたことを検出できない。しかし、putハンドラはputが成功した場合には成功カウンタとカウンタの双方をインクリメントするのに対して、putが成功しなかった場合には成功カウンタをインクリメントしない。したがって、利用者はput関数の完了ハンドリング関数中で両者の値を比較することにより、ロックしたウィンドウへのputが行われたことを検知できる。すなわち、ロックされたウィンドウへputが行われた場合には、成功カウンタの値がカウンタの値よりも小さくなっているはずである。

【0052】(put/getデーモンの実現方法) 次に、本発明のリモートメモリ転送方式におけるput/getデーモンの実現方法の例を図6を参照して説明する。

【0053】put/getデーモンは通常はいずれかのPUのユーザプログラムからのput/get要求待ちでブロック状態にある(601)。put/get関数コールによりput/get要求を受信すると、それがput要求であるかget要求であるかによって分岐し(602)、各々に対応する処理に入る。put要求であった場合には、まず送信されてくるウィンドウ識別子とインクリメント値を読み込んで記憶し(603)、put要求元プログラムにackを返す(604)。次いで、put要求元から送信されてくるメッセージ本体を受信して記憶し(605)、メッセージをすべて受診すると自PU上の利用者プログラムにたいして割り込みを発生する(606)。(この割り込みによって利用者プログラム側ではputハンドラがコールされ、受信待ち状態となる。)デーモンは利用者プログラムに対して、603にて記憶したウィンドウ識別子を送信し、肯

11

定応答 (ack) または否定応答 (nack) を待って処理を分岐する (610)。ack を受け取った場合には、当該ウィンドウはロック状態にはないので、605にて記憶したメッセージ本体を利用者プログラムに送信して処理を終了する。nack を受け取った場合には、当該ウィンドウはロックされているので、メッセージを破棄して処理を終了する。一方602にて get 要求へ分岐した場合には、要求元PUからウィンドウ識別子を受け取り (612)、自PU上の利用者プログラムに割り込みをかける (613)。(この割り込みによって利用者プログラム側では get ハンドラがコールされ、受信待ち状態となる。) デモンは利用者プログラムに対して、612にて記憶したウィンドウ識別子を送信し、肯定応答 (ack) または否定応答 (nack) を待って処理を分岐する (618)。ack を受け取った場合には、当該ウィンドウはロック状態にはないので、利用者プログラムの get 処理ハンドリング関数よりメッセージが送られてくる。これを記憶し、get 要求元に送信して処理を終了する (616, 617)。nack を受け取った場合には、当該ウィンドウはロックされておりメッセージは送られてこないの、get 要求元にエラーコードを返す (619)。

【0054】以上が put/get デモンの実現方法の例である。

【0055】

【発明の効果】本発明のリモートメモリ転送制御方式によれば、メッセージバッシングライブラリを用いた put/get 方式のメッセージ通信において、put/get 用のメモリ領域を利用者プログラム中からロック、アンロックすることが可能となる。これによって利用者は put/get 用のメモリ領域に複数のPUからのデータが上書きされたり、不正なデータが読みだされたりす\*

12

\* ることを防止することができ、put/get 方式によるメッセージ通信をより安全に行なうことができるようになる。

【0056】また、本発明のリモートメモリ転送制御方式によれば、何等かの原因でロックされたメモリ領域にたいして誤った put/get がなされた場合に、これを検出することができる。これによって利用者は、適当なエラーリカバリをプログラム中に組み込むことが可能となり、put/get 方式を用いて信頼性の高いプログラムを作成することができる。

【図面の簡単な説明】

【図1】本発明の実施例の全体構成図。

【図2】ウィンドウ情報テーブルの説明図。

【図3】ウィンドウ生成関数の動作説明図。

【図4】put 関数の動作説明図。

【図5】get 関数の動作説明図。

【図6】put/get デモンプロセスの動作説明図。

【図7】put ハンドラの動作説明図。

【図8】get ハンドラの動作説明図。

【符号の説明】

101, 102... 要素計算機

103, 104... CPU

105, 106... メモリ

107... 通信路

108, 109... オペレーティングシステム

110, 111... 利用者プログラム

112, 113... put/get デモンプロセス

114, 115... メッセージバッシングライブラリ

120, 121... put/get 割り込みハンドラ

122, 123... put/get 用ウィンドウ。

【図2】

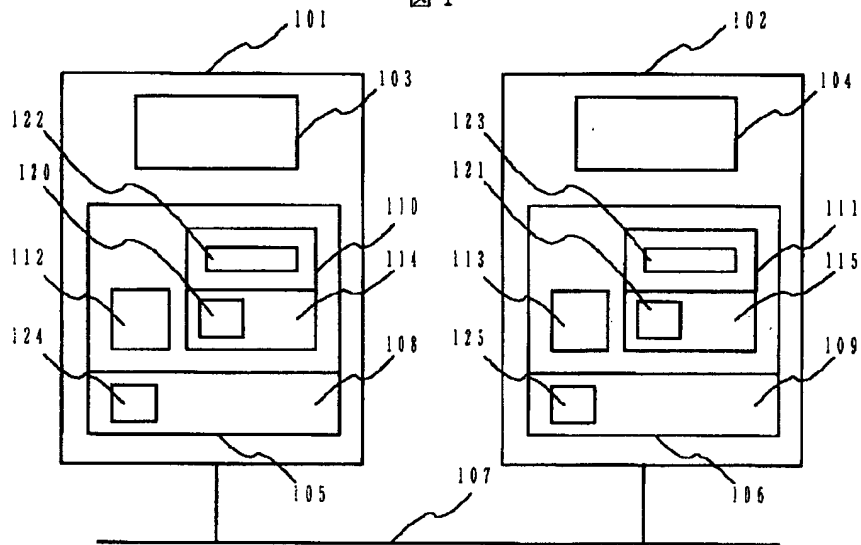
図2

ウィンドウ 識別子	バッファ 先頭 アドレス	バッファ 末尾 アドレス	カウン タ 値	成功 カウン タ 値	カウン タ 上限値	ロック 状態 フラグ	ハン ドラ



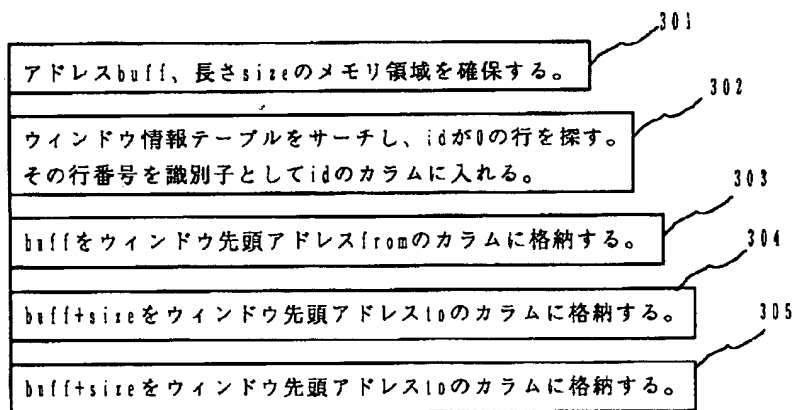
【図1】

図1



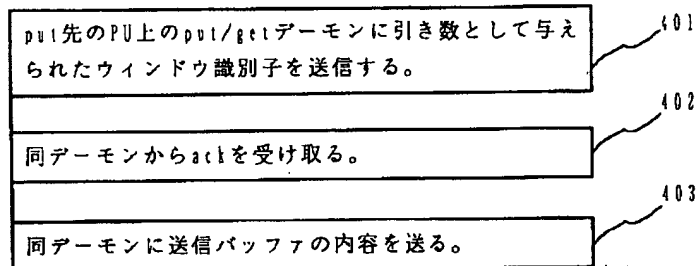
【図3】

図3



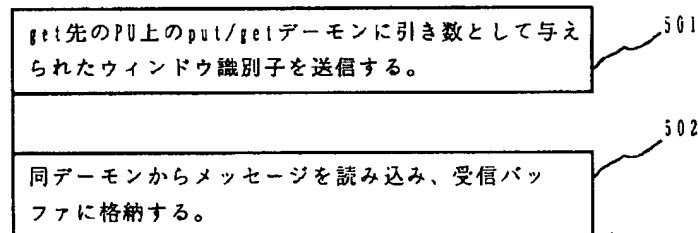
【図4】

図4



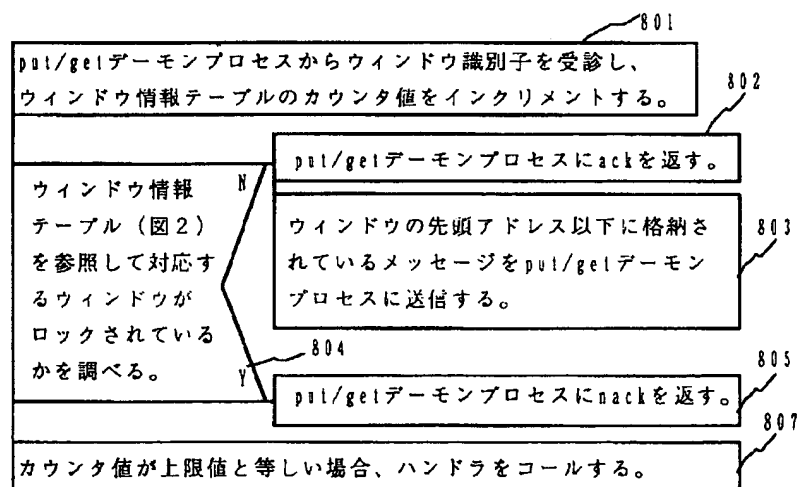
【図5】

図5



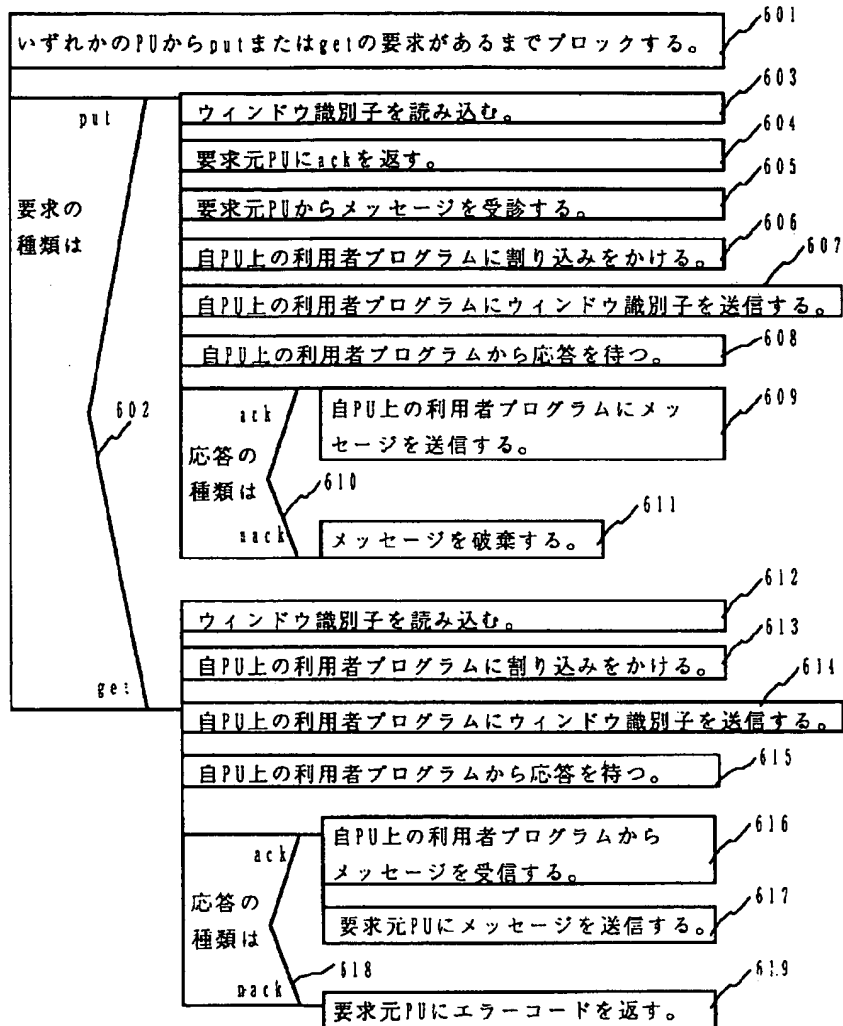
【図8】

図8



【図6】

図 6



【図7】

図7

